Lab 1 – Multiplexers

Shubham Mishra

San Jose State University, College of Engineering,

1 Washington Sq. San Jose, CA 95192

***Abstract-* This lab report is about Multiplexers (MUX) and how their behaviors change depending on the number of inputs and select lines. The experiment was conducted using structural Verilog where it was used to verify the validity of the output equations for 2-1 and 3-1 MUXs.**

***Keywords- Multiplexers, Structural Verilog, 2-1 MUX, 3-1 MUX, logic levels, AND, OR Gates***

I. Introduction

A Multiplexer or MUX is a logic circuit designed to switch one of several input lines through a single output. The MUXs introduced in this lab are the 2-1 MUX, with 2 inputs and 1 output, and a 3-1 MUX with 3 inputs and 1 output.

II. Experiments

A. 2-1 MUX

|  |  |  |  |
| --- | --- | --- | --- |
| Select | A | B | Output |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

*Table 1 -* Truth Table for 2-1 MUX

From Table 1 it can be shown that output is A when the select line is equal to 1. When select is equal to 0, output is B. The equation that can be derived from this table is:

OUT = S’B + SA.

|  |  |
| --- | --- |
| Test Bench  module and\_tb();  reg a,b,s;  wire y;  mux mux\_tb(a,b,y,s);    initial begin  a=0; b=0; s=0;  #2  a=0; b=1; s=0;  #2  a=1; b=0; s=0;  #2  a=1; b=1; s=0;  #2  a=0; b=0; s=1;  #2  a=0; b=1; s=1;  #2  a=1; b=0; s=1;  #2  a=1; b=1; s=1;    $display("a=%b,b=%b,s=%b",a,b,s);  #1  $display("y=%b",y);  end  initial begin  $dumpfile("dump.vcd");  $dumpvars(1);  end  endmodule | Design Bench  module mux(a,b,c,y,s1,s2);    input a,b,c,s1,s2;  wire d,e,f,g,h;  output y;    not(d,s1);  not(e,s2);  and(f,a,d,e);  and(g,b,d,s2);  and(h,c,s1);    or(y,f,g,h);  endmodule |

*Figure 1* – Test and Design Bench for 2-1 MUX

A picture containing chart

Description automatically generated

*Figure 2 – Waveform for 2-1 MUX*

As shown in Figure 2, we can see that when the select line is 0, the output is B which proves the validity of our code.

B. 3-1 MUX

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Select1 | Select2 | A | B | C | Output |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 |

*Table 2 –* Truth table for 3-1 MUX

From Table 2, we can observe that when Select1 and Select2 are 0, the output is A. When Select1 is 0 and Select2 is 1, the output is B. When Select1 is 1 and Select2 is 0, the output is C. When Select1 and Select 2 are both 1, the output is also C. This can be generalized to an equation as follows:

OUT = S1’S2’A + S1’S2B + S1S2’C + S1S2C

|  |  |
| --- | --- |
| module and\_tb();  reg a,b,c,s1,s2;  wire y;  mux mux\_tb(a,b,c,y,s1,s2);    initial begin  s1=0; s2=0; a=0; b=0; c=0;  #2  s1=0; s2=1; a=0; b=0; c=1;  #2  s1=1; s2=0; a=0; b=1; c=0;  #2  s1=1; s2=1; a=0; b=1; c=1;  #2  s1=0; s2=0; a=1; b=0; c=0;  #2  s1=0; s2=1; a=1; b=0; c=1;  #2  s1=1; s2=0; a=1; b=1; c=0;  #2  s1=1; s2=1; a=1; b=1; c=1;    $display("a=%b,b=%b,s1=%b,s2=%b",a,b,s1,s2);  #1  $display("y=%b",y);  end  initial begin  $dumpfile("dump.vcd");  $dumpvars(1);  end  endmodule | module mux(a,b,c,y,s1,s2);    input a,b,c,s1,s2;  wire d,e,f,g,h;  output y;    not(d,s1);  not(e,s2);  and(f,a,d,e);  and(g,b,d,s2);  and(h,c,s1);  or(y,f,g,h);    endmodule |

*Figure 3 –* Test and Design Bench for 3-1 MUX

Timeline

Description automatically generated

*Figure 4 –* Waveform for 3-1 MUX

As shown in Figure 4, we can see that when both S1 = 1 and S2 = 0, the output is C which proves that the code we ran agrees with our truth table.

III. Conclusion

Overall, in this lab after learning about multiplexers for 2-1 and 3-1 circuits, it showed how to design a Verilog program using a circuit diagram and use truth tables to verify the validity of the program.